Software Requirements Specification

for

<Project>

Prepared by <author>

<organization>

<date created>

.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

1.2 Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

1.3 Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

1.4 Project Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.>

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2. Overall Description

2.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Features

<Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Details will be provided in Section 3, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.>

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

3. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

3.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

3.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

3.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

3.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:
REQ-2:

3.2 System Feature 2 (and so on)

4. External Interface Requirements

4.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

4.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

4.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

4.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: Issues List

< This is a dynamic list of the open requirements issues that remain to be resolved, including TBDs, pending decisions, information that is needed, conflicts awaiting resolution, and the like.>

About CA 1: *Gamification approach is used for the CA 1 on basis of milestones mentioned.*

Modules submission weekly Detail	**Rubrics**	The following **Badges** will be given to the students
Week 5: SRS Document as per **IEEE SRS format** **Week 7:** System Design **Week 10:** Test Cases **Week 12:** Final Submission	• SRS Document as per IEEE SRS format: 30% • System Design: 30% • Test Cases: 20% • Viva: 20%	**Initiator** – Awarded for selecting the project topic and identifying stakeholders. **Analyzer** – For completing stakeholder analysis and understanding user needs. **Innovator** – For proposing creative and unique functional/non-functional requirements. **Visualizer** – For crafting accurate and visually clear diagrams (e.g., UML, DFD). **Strategist** – For completing the feasibility study with well-researched insights. **Timekeeper** – For submitting the SRS on or before the deadline. **Champion** – For overall excellence in completing the SRS with the highest score. **Collaborator** – For contributing effectively to team discussions and peer reviews.

Hint

Example of a **University Management System (UMS)**:

---

1. Introduction

1.1 Purpose:

The purpose of the UMS is to provide a unified platform for managing all university operations, including **student registrations**, **fee payments**, **course scheduling**, and **faculty management**. It **aims to** *automate administrative tasks, enhance efficiency, and improve user experience for students, faculty, and administrators.*

1.2 Document Conventions:

Document Conventions refers to the **standards, formatting, and guidelines** followed in the document to ensure consistency, clarity, and ease of understanding for the readers. It specifies how information is organized, how terms are defined, and the conventions for presenting content.

Example for UMS:

- **Formatting Guidelines**: Bold text for section titles, italic for emphasis, and numbered lists for steps or procedures.
- **Terminology**: Terms like "Student," "Faculty," and "Administrator" are consistently used throughout the document.

This document follows standard conventions such as:

- Keywords like **"must"**, **"should"**, and **"may"** denote mandatory, recommended, and optional requirements respectively.
- UML diagrams are **used for system design**.
- Tables and bullet points are **used for clarity**.

1.3 Intended Audience and Reading Suggestions:

The document is intended for:

- **Developers**: To understand system design and implement features.
- **Project Managers**: For tracking progress and ensuring all requirements are met.
- **University Stakeholders**: To validate the system's functionality.

1.4 Product Scope:

The **UMS will handle** student enrolment, fee payment, attendance tracking, exam scheduling, and grading. The system will be accessible via a web portal and mobile application.

1.5 References:

Include references to similar projects, guidelines, or frameworks used e.g.

- IEEE 830-1998 for Software Requirements Specifications.
- University policies for academic operations.

---

2. Overall Description

2.1 Product Perspective:

UMS integrates with existing university systems such as the library management system, financial systems, and learning management systems. **It acts as a central hub for managing academic and administrative processes.**

2.2 Product Functions: provide a high-level overview of the functional requirements. This is less detailed but serves as a summary.

- **Student Module**: Registration, profile management, and attendance tracking.
- **Faculty Module**: Course scheduling, attendance marking, and grade submissions.
- **Admin Module**: Management of courses, departments, and user roles.

2.3 User Classes and Characteristics:

- **Students**: Require a user-friendly interface for registrations and accessing academic records.
- **Faculty**: Need tools for managing schedules and grades.
- **Administrators**: Require advanced control for system configuration and report generation.

2.4 Operating Environment:

- **Web Application**: Accessible via modern web browsers like Chrome and Firefox.
- **Mobile App**: Available for Android and iOS.
- The system will be hosted on a cloud platform for scalability.

2.5 Design and Implementation Constraints:

These are limitations or rules the system must follow while being designed or developed.

- **Must comply with GDPR for data privacy:** The system must follow strict rules to protect users' personal data, as required by the General Data Protection Regulation (GDPR). For

example, students' information should not be shared without their consent, and their data must be stored securely.

- **Should support a minimum of 5,000 concurrent users:** The system should be capable of handling at least 5,000 users logged in and using the platform at the same time without crashing or slowing down.
- Use of open-source technologies like **MySQL** and **Java Spring Framework**.

2.6 User Documentation:

The system will include user manuals, FAQs, and video tutorials for ease of use.

2.7 Assumptions and Dependencies:

- Reliable internet access is assumed for users.
- Integration with third-party systems (e.g., payment gateways) depends on their availability.

---

3. External Interface Requirements

3.1 User Interfaces:

- A **dashboard** for students to view grades and schedules.
- Faculty interface for attendance and exam results.
- Admin panel for system configuration.

3.2 Hardware Interfaces:

- The system should support biometric devices for attendance tracking.
- Should work on PCs, laptops, tablets, and smartphones.

3.3 Software Interfaces:

- Integration with learning management systems (e.g., Moodle).
- Payment gateways for online fee transactions.

3.4 Communications Interfaces:

- Email notifications for fee receipts and announcements.
- Push notifications for deadlines and results via the mobile app.

---

4. System Features

Functional requirements are included in the **"System Features"** section of the document.

4.1 System Feature 1:

Student Registration:
The system must allow students to register for courses, view their academic details, and access fee payment options.

4.2 System Feature 2:

Attendance Tracking:
The faculty should be able to mark attendance online, and students can view their attendance status.

4.3 System Feature 3:

Fee payment processing

4.4 System Feature 4:

Grade management

---

5. Other Non-functional Requirements

5.1 Performance Requirements:

- The system must handle **5,000 concurrent users** without performance degradation.
- Response time should not exceed **2 seconds** for any transaction.

5.2 Safety Requirements:

- Ensure data backups are taken daily to avoid data loss.
- Restrict unauthorized access to critical modules.

5.3 Security Requirements:

- Use **256-bit encryption** for all sensitive data.
- Multi-factor authentication for admin accounts.

5.4 Software Quality Attributes:

- **Scalability**: The system should handle increased user loads during peak admission periods.
- **Reliability**: Uptime should be **99.9%** annually.

5.5 Business Rules:

- Only enrolled students can access the UMS.
- Students must clear fee dues to register for new courses.

---

6. Other Requirements

Any additional requirements such as data import/export tools or APIs for third-party integration will be detailed here.

Appendices

- **Appendix A: Glossary**: Define terms like "UMS," "LMS," "biometric attendance," etc.
- **Appendix B: Analysis Models**: Include diagrams like use case diagrams and data flow diagrams. (note: their inclusion depends on the level of detail and format of the document.)

  Data Flow Diagrams (DFDs): To represent how data moves through the system.

  Use Case Diagrams: To visualize interactions between users (actors) and the system.
- **Appendix C: To Be Determined List**: Document any incomplete or pending requirements.